A. Projected Sinkhorn

Here, we give a brief description of the approximate projection method proposed by Wong et al. (2019). The projection of a (normalized) vector $\mathbf{w}$ to the Wasserstein ball centered at (normalized) $\mathbf{x}$ of radius of $\epsilon = \delta$ can be written as:

$$\begin{aligned} &\underset{\mathbf{z},\Pi\geq 0}{\text{minimize}} \quad \frac{1}{2}\left\|\mathbf{w}-\mathbf{z}\right\|_2^2 \\ &\text{subject to } \Pi\mathbf{1}=\mathbf{x},\ \Pi^\top\mathbf{1}=\mathbf{z}, \langle\Pi, C\rangle \leq \epsilon. \end{aligned}$$

The above objective is not strongly convex in Π, but can be made strongly convex by adding an entropic regularization:

$$\begin{aligned} &\underset{\mathbf{z},\Pi\geq 0}{\text{minimize}} \quad \frac{1}{2}\left\|\mathbf{w}-\mathbf{z}\right\|_2^2 + \gamma\sum_{i=1}^n\sum_{j=1}^n \Pi_{ij}\log\Pi_{ij} \\ &\text{subject to } \Pi\mathbf{1}=\mathbf{x},\ \Pi^\top\mathbf{1}=\mathbf{z}, \langle\Pi, C\rangle \leq \epsilon. \end{aligned} \tag{20}$$

The parameter $\gamma > 0$ is the entropic regularization constant. Projected Sinkhorn solves (20) through block-coordinate maximization on the dual problem of (20).

A.1. Analysis of Approximation Error in Projected Sinkhorn

To ensure small approximation error in (20), the scale of entropic regularization term should be at least much smaller than the quadratic term:

$$\gamma\sum_{i=1}^n\sum_{j=1}^n \Pi_{ij}\log\Pi_{ij} \ll \frac{1}{2}\left\|\mathbf{w}-\mathbf{z}\right\|_2^2.$$

Otherwise, the objective (20) is dominated by the entropic regularization. However, in practice, it is not always guaranteed, especially when $\mathbf{w}$ is an interior point of the constraint in (20).

Consider an simple example where $\mathbf{w} = \mathbf{x} = (\frac{1}{n}, \frac{1}{n}, \cdots \frac{1}{n})^\top$. In that case, the quadratic term $\frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2$ should be as small as zero, since we can let $\mathbf{z} = \mathbf{x}$. However, if $\mathbf{z} = \mathbf{w} = \mathbf{x}$, then Π could be a diagonal matrix $\operatorname{diag}\left(\frac{1}{n}, \frac{1}{n}, \cdots \frac{1}{n}\right)$ (or more generally, $\frac{1}{n}P$, where P is a permutation matrix). Thus, the entropic term becomes

$$\sum_{i=1}^n\sum_{j=1}^n \Pi_{ij}\log\Pi_{ij} = -\log n, \tag{21}$$

reaching its *maximum*. The entropic regularization somewhat conflicts with the the quadratic term. Notice that the scale of (21) is much larger than $\frac{1}{2}\|\mathbf{w}-\mathbf{x}\|_2^2$ (which is supposed to be very close to zero), especially when the dimension n is large. Thus, the objective (20) may be dominated by the entropic regularization and solving the projection step accurately requires very small γ.

We make two additional remarks. First, notice that the scale of (21) increases as n grows, which requires smaller γ to balance the quadratic term and entropic regularization. This gives the intuition that projected Sinkhorn needs smaller γ in higher dimensional spaces, which is observed in experiments.

Second, the key aspect of the above argument is that $\mathbf{w}$ is relatively close to $\mathbf{x}$, *e.g.* $\mathcal{W}(\mathbf{w}, \mathbf{x}) \leq \epsilon$, such that the quadratic term is so small hence dominated by the entropic regularization. In the case where $\mathbf{w}$ is very far away from $\mathbf{x}$, this argument does not hold anymore. We believe this explains why large step sizes strengthen the attack when using PGD with projected Sinkhorn in experiments. However, PGD with large step sizes tend to be unstable and may not converge to a good solution.

A.2. Toy Experiment in Table 2

Entries of $\mathbf{a}$ and $\mathbf{b}$ are sampled from a uniform distribution in $[0, 1]^{400}$ independently. After sampling, both vectors are normalized to ensure that the pixel mass summations are exactly 1. We reshape $\mathbf{a}$ and $\mathbf{b}$ to $\mathbb{R}^{20\times 20}$ and view them as images in order to use the procedure of Wong et al. (2019). The cost matrix is induced by Euclidean norm between pixel indices with 5×5 local transportation plan.

We use projected gradient descent and Frank-Wolfe to compute the projection by solving the following "reparametrized" projection problem w.r.t. the coupling matrix:

$$\begin{aligned} &\underset{\Pi}{\text{minimize}} \ \ \frac{1}{2}\|\Pi^\top \mathbf{1} - \mathbf{b}\|_2^2 \\ &\text{subject to } \ \Pi \geq 0, \Pi\mathbf{1} = \mathbf{a}, \langle \Pi, C \rangle \leq \epsilon. \end{aligned}$$

The problem is equivalent to the Euclidean projection in the image space and is convex in Π. For PGD, we use step size 0.05. For Frank-Wolfe, we use $\gamma = 10^{-3}$ and the default decay schedule $\frac{2}{t+1}$. We let both algorithms run for sufficiently many iterations in order to converge to high precision solutions.

B. Recommended Stopping Criterion for Bisection Search

When the derivative of the dual objective approaches zero, *i.e.*, $\langle \Pi, C \rangle - \delta \approx 0$, the comparison between $\langle \Pi, C \rangle - \delta$ and 0 is getting numerically unstable. Thus, we recommend stopping the bisection method when either the derivative is close to zero, or the gap between the lower bound l and the upper bound u is relatively small.

We recommend using an upper bound u to recover the coupling Π. Since an upper bound u always has a negative derivative, thus the transportation cost constraint $\langle \Pi, C \rangle \leq \delta$ is always satisfied.

We highlight that the bisection method converges very fast in practice, since it shrinks the interval by a factor of 2 in every iteration. Thus, it determines the next 3 digits of $\lambda^\star$ after the decimal point after every 10 iterations.

For a concrete stopping criterion in our experiment, please refer to Appendix F.2.

C. Dykstra's Projection

Algorithm 2: Dykstra's Projection Algorithm

Input: $G \in \mathbb{R}^{n \times n}$, two convex sets $\mathcal{C}_s$ and $\mathcal{C}_h$
Output: The projection of G to $\mathcal{C}_s \cap \mathcal{C}_h$

$\Pi_h^{(0)} = G$
$I_s^{(0)} = I_h^{(0)} = O$
for $t = 0, 1, \ldots,$ maxiter **do**
  $\Pi_s^{(t+1)} = \mathrm{Proj}_{\mathcal{C}_s}\left(\Pi_h^{(t)} - I_s^{(t)}\right)$
  $I_s^{(t+1)} = \Pi_s^{(t+1)} - \Pi_h^{(t)} + I_s^{(t)}$
  $\Pi_h^{(t+1)} = \mathrm{Proj}_{\mathcal{C}_h}\left(\Pi_s^{(t+1)} - I_h^{(t)}\right)$
  $I_h^{(t+1)} = \Pi_h^{(t+1)} - \Pi_s^{(t+1)} + I_h^{(t)}$
return $\Pi_s^{(t+1)}$

Consider the projection of $G \in \mathbb{R}^{n \times n}$ to the intersection of $\mathcal{C}_s$ and $\mathcal{C}_h$, where $\mathcal{C}_s = \{\Pi \in \mathbb{R}^{n \times n} : \Pi \geq 0,\ \Pi \mathbf{1} = \mathbf{x}\}$ and $\mathcal{C}_h = \{\Pi \in \mathbb{R}^{n \times n} : \langle \Pi, C \rangle \leq \delta\}$. Dykstra's algorithm, applying to these two convex sets, is presented in Algorithm 2. Intuitively, Dykstra's algorithm projects G alternatively to $\mathcal{C}_s$ and $\mathcal{C}_h$ in each iteration. Notice that before projecting to $\mathcal{C}_s$ (or $\mathcal{C}_h$), the increment of the last iteration $I_s^{(t)}$ (or $I_h^{(t)}$) is subtracted from $\Pi_h^{(t)}$ (or $\Pi_s^{(t+1)}$). These increments play a crucial role in the convergence of Dykstra's algorithm. It has been shown that both $\Pi_s^{(t)}$ and $\Pi_h^{(t)}$ converge to the projection of G onto $\mathcal{C}_s \cap \mathcal{C}_h$ (Dykstra, 1983; Boyle & Dykstra, 1986).

In order to implement Dykstra's algorithm, we need two subroutines to compute the projection onto $\mathcal{C}_s$ and $\mathcal{C}_h$ respectively. The projection onto $\mathcal{C}_h$ admits a closed form expression:

$$\mathrm{Proj}_{\mathcal{C}_h}(\Pi) = \Pi - \frac{\max\{\langle \Pi, C \rangle - \delta, 0\}}{\|C\|_{\mathrm{F}}^2} C.$$

The projection onto $\mathcal{C}_s$ has an algorithm running in $O(n^2 \log n)$ time, by projecting each row of G to a simplex. For the simplex projection algorithm, we direct readers to (Duchi et al., 2008).

C.1. Toy Experiment for Dykstra's Algorithm

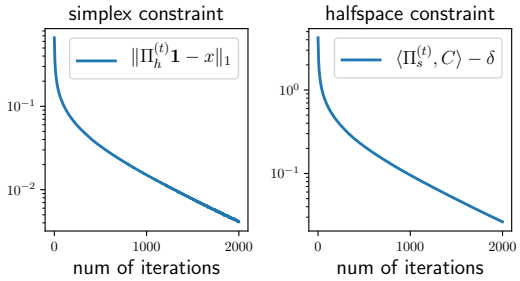


Figure 7: Convergence of Dykstra's algorithm. The violation of simplex constraint and halfspace constraint of $\Pi_h^{(t)}$ and $\Pi_s^{(t)}$ (the iterates produced by Dykstra's algorithm) gradually decrease to zero, but at a slow rate.

We randomly sample a vector $\mathbf{x} \in \mathbb{R}^{100}$ and a coupling $\Pi \in \mathbb{R}^{100 \times 100}$ (both from a uniform distribution in a hypercube). We normalize $\mathbf{x}$ and Π. We then project Π to $\mathcal{C}_s \cap \mathcal{C}_h$. We set $\delta = 1$ and the cost matrix C is the same as the one in §6 (we reshape $\mathbf{x}$ into a 10×10 matrix and view it as an image). The convergence plots are shown in Figure 7. Dykstra's algorithm does converge, but at a slow rate.

D. Toy Example: Failure of Dual Linear Minimization without Entropic Regularization

This section presents a simple example to demonstrate the failure of dual LMO without adding entropic regularization.

Let $\delta = 0.5$, $\mathbf{x} = (1, 0)^\top$. Let

$$G = \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Let

$$\Pi = \begin{pmatrix} \Pi_{11} & \Pi_{12} \\ \Pi_{21} & \Pi_{22} \end{pmatrix}.$$

Then the primal linear program is

$$\begin{aligned} &\underset{\Pi_{11}, \Pi_{12} \geq 0}{\text{minimize}} \;\; \Pi_{11} - \Pi_{12} \\ &\text{subject to } \Pi_{11} + \Pi_{12} = 1, \Pi_{12} \leq 0.5 \end{aligned}$$

It is easy to check that the solution is

$$\Pi^\star = \begin{pmatrix} 0.5 & 0.5 \\ 0 & 0 \end{pmatrix}.$$

The dual linear program is

$$\underset{\lambda \geq 0}{\text{maximize}} -\frac{1}{2}\lambda + \min\{1, -1 + \lambda\}$$

It is easy to see that the dual problem has a unique solution $\lambda^\star = 2$. Now we try to use the following condition to recover the primal solution:

$$\Pi^\star \in \underset{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}}{\operatorname{argmin}} \; \langle \Pi, G + \lambda^\star C \rangle - \lambda^\star \delta, \tag{22}$$

which is equivalent to

$$\Pi^\star \in \underset{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}}{\operatorname{argmin}} \; \left\langle \Pi, \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix} \right\rangle. \tag{23}$$

But it turns out that any Π of the form

$$\Pi(\alpha) = \begin{pmatrix} \alpha & 1 - \alpha \\ 0 & 0 \end{pmatrix},$$

where $0 \leq \alpha \leq 1$, is a minimizer. By varying α, $\Pi(\alpha)$ can be suboptimal ($\alpha = 0$), optimal ($\alpha = 0.5$) or even infeasible ($\alpha = 1$). Thus, $\Pi^\star$ cannot be recovered by only considering the stationary condition.

Of course it is possible to combine (22) with other KKT conditions (specifically, complementary slackness and primal feasibility) to obtain one of the primal solutions. Particularly, in the above example, (22) along with the complementary slackness determines the unique primal solution $\Pi^\star$. However, there are still two issues. The first issue is that in more general cases, doing so requires solving a linear system whose variables are from a subset of Π, which could be GPU unfriendly. More critically, the above solution is numerically unstable. Suppose that there is a slight numerical inaccuracy due to floating point precision, such that (23) becomes

$$\Pi^\star \in \underset{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}}{\operatorname{argmin}} \; \left\langle \Pi, \begin{pmatrix} 1 + \xi & 1 - \xi \\ 2 & 0 \end{pmatrix} \right\rangle. \tag{24}$$

for some small constant $\xi > 0$. Now solving (24) gives

$$\Pi(1) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},$$

which is infeasible.

E. Proofs

This section presents all proofs in the paper. Recall that without loss of generality we assume that all entries in the cost matrix C are nonnegative and only diagonal entries of C are zeros. All proofs below assume it implicitly.

E.1. Dual Projection

Proposition 1. *The dual of* (4) *is*

$$\underset{\lambda \geq 0}{\text{maximize}}\ g(\lambda), \quad \textit{where} \tag{5}$$

$$g(\lambda) = \min_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \tfrac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 + \lambda\left(\langle \Pi, C\rangle - \delta\right). \tag{6}$$

In addition, the derivative of $g(\lambda)$ at a point $\lambda = \tilde{\lambda}$ is

$$g'(\tilde{\lambda}) = \langle \tilde{\Pi}, C\rangle - \delta, \quad \textit{where} \tag{7}$$

$$\tilde{\Pi} = \operatorname*{argmin}_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \|\Pi - G + \tilde{\lambda} C\|_{\mathrm{F}}^2. \tag{8}$$

Both $g(\lambda)$ and $g'(\lambda)$ can be evaluated in $O(n^2 \log n)$ time deterministically for any given λ.

Proof. Introducing the Lagrange multiplier $\lambda \geq 0$ for the constraint $\langle \Pi, C\rangle \leq \delta$, we arrive at the following dual problem

$$\underset{\lambda \geq 0}{\text{maximize}}\ g(\lambda),$$

where

$$g(\lambda) = \min_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \frac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 + \lambda\left(\langle \Pi, C\rangle - \delta\right).$$

We complete the square in the inner problem, which leads to

$$g(\lambda) = \min_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \frac{1}{2}\|\Pi - (G - \lambda C)\|_{\mathrm{F}}^2 - \frac{1}{2}\lambda^2 \|C\|_{\mathrm{F}}^2 + \lambda\langle G, C\rangle - \lambda\delta.$$

Notice that the constraint in the minimization is independent for each row of Π. Thus, it can be reduced to a simplex projection for each row of $G - \lambda C$, which can be solved in $O(n^2 \log n)$ time.

By Danskin's theorem, g is differentiable and the derivative is

$$g'(\tilde{\lambda}) = \langle \tilde{\Pi}, C\rangle - \delta,$$

given the solution $\tilde{\Pi}$ to the minimization problem. □

Before proving Proposition 2, we first prove Lemma 1, which characterizes the solution of simplex projection in a special case. Intuitively, the projection of a vector to a simplex is very sparse if one of its entries is much larger than the others.

Lemma 1. *Consider the following projection of a vector $\mathbf{v}$ to a simplex:*

$$\begin{aligned} &\underset{\mathbf{w} \in \mathbb{R}^n}{\text{minimize}} \quad \|\mathbf{w} - \mathbf{v}\|_2^2 \\ &\text{subject to} \ \sum_{i=1}^n w_i = z,\ w_i \geq 0, \end{aligned}$$

where $z > 0$. Suppose that there exists i such that $v_i \geq v_j + z$ for all $j \neq i$. Then the solution is $\mathbf{w}^\star = (0, \cdots 0, z, 0, \cdots, 0)^\top$, where the only nonzero entry is $w_i^\star = z$.

Proof. A careful analysis of the simplex projection algorithm (Duchi et al., 2008) would give a proof. Here, we give an alternative simple proof that does not rely on the algorithm. Assume to the contrary that there exists $j \neq i$ such that $w_j^\star > 0$ hence also $w_i^\star < z$. We construct another feasible point $\hat{\mathbf{w}}$ by

$$\begin{aligned}\hat{w}_i &= w_i^\star + w_j^\star \\ \hat{w}_j &= 0 \\ \hat{w}_k &= w_k^\star \ \forall k \neq i, k \neq j.\end{aligned}$$

Comparing the objective value of $\mathbf{w}^\star$ and $\hat{\mathbf{w}}$, we have

$$\begin{aligned}\|\mathbf{w}^\star - \mathbf{v}\|_2^2 - \|\hat{\mathbf{w}} - \mathbf{v}\|_2^2 &= (w_i^\star - v_i)^2 + (w_j^\star - v_j)^2 - (w_i^\star + w_j^\star - v_i)^2 - (0 - v_j)^2 \\ &= 2w_j^\star(v_i - v_j - w_i^\star) \\ &> 2w_j^\star(v_i - v_j - z) \\ &\geq 0.\end{aligned}$$

$\hat{\mathbf{w}}$ has even smaller objective value than $\mathbf{w}^\star$, contradicting the optimality of $\mathbf{w}^\star$. Thus all $w_j^\star = 0$ for all $j \neq i$, which finishes the proof. □

Proposition 2. *The dual solution $\lambda^\star$ of (5) satisfies*

$$0 \leq \lambda^\star \leq \frac{2\left\|\mathrm{vec}(G)\right\|_\infty + \left\|\mathbf{x}\right\|_\infty}{\min_{i \neq j}\{C_{ij}\}}. \tag{9}$$

Proof. By Danskin's theorem the dual problem (5) is differentiable in λ. Moreover, for any given $\tilde{\lambda}$, suppose the solution to the minimization is $\tilde{\Pi}$, then the gradient w.r.t. $\tilde{\lambda}$ is $\langle \tilde{\Pi}, C\rangle - \delta$.

Consider the i-th row of $G - \lambda C$. Assume on the contrary that

$$\lambda > \frac{2\left\|\mathrm{vec}(G)\right\|_\infty + \left\|\mathbf{x}\right\|_\infty}{\min_{i \neq j}\{C_{ij}\}}.$$

Then, for all $i \neq j$ we have (recall that $C_{ii} = 0$)

$$G_{ii} = G_{ii} - \lambda C_{ii} > G_{ij} - \lambda C_{ij} + x_i.$$

The condition in Lemma 1 is satisfied. A projection of $G - \lambda C$ results in a diagonal matrix $\tilde{\Pi}$. Thus

$$\begin{aligned}g'(\tilde{\lambda}) &= \langle \tilde{\Pi}, C\rangle - \delta \\ &= \sum_{i=j} \tilde{\Pi}_{ij} C_{ij} + \sum_{i \neq j} \tilde{\Pi}_{ij} C_{ij} - \delta \\ &= -\delta.\end{aligned}$$

The derivative is strictly negative, hence $\tilde{\lambda}$ is suboptimal, which finishes the proof.

□

Proposition 3. *The primal solution $\Pi^\star$ and the dual solution $\lambda^\star$ satisfies*

$$\Pi^\star = \operatorname*{argmin}_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \|\Pi - G + \lambda^\star C\|_{\mathrm{F}}^2,$$

thus $\Pi^\star$ can be computed in $O(n^2 \log n)$ time given $\lambda^\star$.

Proof. This is a direct implication of KKT conditions. □

E.2. Dual Linear Minimization Oracle without Entropic Regularization

Proposition 4. *The dual problem of* (12) *is*

$$\underset{\lambda \geq 0}{\text{maximize}} \;\; -\lambda\delta + \sum_{i=1}^{n} x_i \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big). \tag{13}$$

Proof. Introducing the Lagrange multiplier $\lambda \geq 0$ for the constraint $\langle \Pi, C \rangle \leq \delta$, we arrive at the following dual problem

$$\underset{\lambda \geq 0}{\text{maximize}} \; g(\lambda),$$

where

$$\begin{aligned} g(\lambda) &= \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \; \langle \Pi, H \rangle + \lambda \left(\langle \Pi, C \rangle - \delta \right) \\ &= \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \; \langle \Pi, H + \lambda C \rangle - \lambda\delta \\ &= -\lambda\delta + \sum_{i=1}^{n} x_i \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big). \end{aligned}$$

The last equality uses the fact that the constraints are independent for each row, thus the minimization is separable. □

Proposition 5. *The dual solution* $\lambda^\star$ *of* (13) *satisfies*

$$0 \leq \lambda^\star \leq \frac{2 \left\| \mathrm{vec}(H) \right\|_\infty}{\min_{i \neq j} \{C_{ij}\}}. \tag{14}$$

Proof. A key observation is that the dual objective is a piece-wise linear function w.r.t. λ. We can roughly estimate the range of the maximizer, by analyzing the slope of this function.

Suppose

$$\lambda > \frac{2 \left\| \mathrm{vec}(H) \right\|_\infty}{\min_{i \neq j} \{C_{ij}\}}. \tag{25}$$

Then for all $i \neq j$, we have

$$\lambda C_{ij} > H_{ii} - H_{ij},$$

which implies $H_{ii} + \lambda C_{ii} < H_{ij} + \lambda C_{ij}$ for all $i \neq j$ (recall that $C_{ii} = 0$). Thus

$$\begin{aligned} g(\lambda) &= -\lambda\delta + \sum_{i=1}^{n} x_i \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big) \\ &= -\lambda\delta + \sum_{i=1}^{n} x_i \left(H_{ii} + \lambda C_{ii} \right) \\ &= -\lambda\delta + \sum_{i=1}^{n} x_i H_{ii} \end{aligned}$$

is a linear function with negative slope. Thus, any λ satisfies (25) cannot be a dual solution, which completes the proof. □

E.3. Dual Linear Minimization Oracle with Dual Entropic Regularization

Proposition 6. *The dual problem of* (16) *is*

$$\begin{aligned} \underset{\lambda \geq 0}{\text{maximize}} \; & -\lambda\delta + \gamma \sum_{i : x_i > 0} x_i \log x_i \\ & - \gamma \sum_{i=1}^{n} x_i \log \sum_{j=1}^{n} \exp\left(-\frac{H_{ij} + \lambda C_{ij}}{\gamma} \right). \end{aligned} \tag{17}$$

Proof. Introducing the Lagrange multiplier $\lambda \geq 0$ for the constraint $\langle \Pi, C \rangle \leq \delta$, we arrive at the following dual problem

$$\underset{\lambda \geq 0}{\text{maximize}} \; g(\lambda),$$

where

$$g(\lambda) = \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \langle \Pi, H \rangle + \gamma \sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} \log \Pi_{ij} + \lambda(\langle \Pi, C \rangle - \delta) \tag{26}$$

$$= \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} -\lambda\delta + \langle \Pi, H + \lambda C \rangle + \gamma \sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} \log \Pi_{ij}. \tag{27}$$

Without loss of generality, we assume that x_i is strictly positive for all i. Otherwise, $x_i = 0$ implies $\Pi_{ij} = 0$ for all j, thus the i-th row of Π does not even appear in the minimization.

Notice that the inner minimization in (27) is separable, since the constraint on Π is independent for each row. For each row, the minimization is equivalent to a Kullback–Leibler projection to a simplex, which admits a closed form. For the sake of completeness, we give a derivation here. For the i-th row,

$$\begin{aligned}
\sum_{j=1}^{n} \Pi_{ij} \left(H_{ij} + \lambda C_{ij}\right) + \gamma \sum_{j=1}^{n} \Pi_{ij} \log \Pi_{ij} &= \gamma \sum_{j=1}^{n} \Pi_{ij} \log \frac{\Pi_{ij}}{\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)} \\
&= \gamma \sum_{j=1}^{n} \Pi_{ij} \left(\log \frac{\Pi_{ij}/x_i}{\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)/a} + \log x_i - \log a \right) \\
&= \gamma \sum_{j=1}^{n} \Pi_{ij} \log \frac{\Pi_{ij}/x_i}{\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)/a} + \gamma x_i \left(\log x_i - \log a\right) \\
&\geq \gamma x_i \left(\log x_i - \log a\right),
\end{aligned}$$

where $a = \sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)$ is a normalization constant. The last inequality holds if and only if

$$\Pi_{ij} = x_i \frac{\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)}.$$

Plugging in the above expression finishes the proof. □

Proposition 7. *The primal solution $\Pi^\star$ and the dual solution $\lambda^\star$ satisfy*

$$\Pi^\star_{ij} = x_i \cdot \frac{\exp\left(-\frac{H_{ij}+\lambda^\star C_{ij}}{\gamma}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda^\star C_{ij}}{\gamma}\right)}. \tag{18}$$

Proof. This is a direct implication of KKT conditions. See the KL projection derivation in the proof of Proposition 6 for a detailed explanation. □

Proposition 8. *The dual solution $\lambda^\star$ of (17) satisfies*

$$0 \leq \lambda^\star \leq \frac{\left[2 \left\|\mathrm{vec}(H)\right\|_\infty + \gamma \log\left(\frac{1}{\delta} \mathbf{x}^\top C \mathbf{1}\right)\right]_+}{\min_{i \neq j} \{C_{ij}\}}. \tag{19}$$

Proof. To begin with, we have the following bound on the derivative:

$$\begin{aligned}
g'(\lambda) &= -\delta + \sum_{i=1}^{n} x_i \frac{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right) C_{ij}}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)} \\
&= -\delta + \sum_{i=1}^{n} x_i \frac{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right) C_{ij}}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right)} \\
&= -\delta + \sum_{i=1}^{n} x_i \frac{\sum_{j\neq i} \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right) C_{ij}}{1+\sum_{j\neq i} \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right)} \\
&\leq -\delta + \sum_{i=1}^{n} \sum_{j\neq i} x_i \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right) C_{ij}.
\end{aligned}$$

The first equality uses translation invariance property of softmin function. The last inequality uses the fact that $C_{ii} = 0$. Notice that

$$\lambda > \frac{\left[2\left\|\mathrm{vec}(H)\right\|_\infty + \gamma \log\left(\frac{1}{\delta}\mathbf{x}^\top C\mathbf{1}\right)\right]_+}{\min_{i\neq j}\{C_{ij}\}}$$

implies

$$\begin{aligned}
\lambda C_{ij} &> 2\left\|\mathrm{vec}(H)\right\|_\infty + \gamma \log\left(\frac{1}{\delta}\mathbf{x}^\top C\mathbf{1}\right) \\
&\geq H_{ii} - H_{ij} + \gamma \log\left(\frac{1}{\delta}\mathbf{x}^\top C\mathbf{1}\right),
\end{aligned}$$

for all $i \neq j$. Thus, we have

$$\exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right) < \frac{\delta}{\mathbf{x}^\top C\mathbf{1}}$$

for all $i \neq j$. Plug it back to $g'(\lambda)$. We have

$$\begin{aligned}
g'(\lambda) &< -\delta + \frac{\delta}{\mathbf{x}^\top C\mathbf{1}} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} x_i C_{ij} \\
&= -\delta + \frac{\delta}{\mathbf{x}^\top C\mathbf{1}} \cdot \mathbf{x}^\top C\mathbf{1} \\
&= 0.
\end{aligned}$$

The derivative is strictly negative hence λ cannot be optimal, which concludes the proof. □

F. Further Experimental Details

In this section, we present further experimental details as well as some implementation details.

F.1. Models

Our MNIST and CIFAR-10, models are taken from (Wong et al., 2019). The MNIST model is a convolutional network with ReLU activations which achieves 98.89% clean accuracy. The CIFAR-10 model is a residual network with 94.76% clean accuracy. The ImageNet model is a ResNet-50 pretrained neural network, downloaded from PyTorch models subpackage, which achieves 72.0% top-1 clean accuracy on the first 100 samples from the validation set. All experiments are run on a single P100 GPU.

F.2. Stopping Criteria of Projection and Linear Minimization Step

Stopping criterion of projected Sinkhorn Denote $\text{obj}^{(t)}$ as the dual objective value of projected Sinkhorn in t-th iteration, we stop the algorithm upon the following condition is satisfied:

$$|\text{obj}^{(t+1)} - \text{obj}^{(t)}| \leq 10^{-4} + 10^{-4} \cdot \text{obj}^{(t)},$$

which is also used by Wong et al. (2019).

Stopping criterion of dual projection and dual LMO Both dual projection and dual LMO use the bisection method to solve dual problems. Bisection is terminated upon either

$$u - l \leq 10^{-4} \text{ or } |g'(\tilde{\lambda})| \leq 10^{-4}$$

This condition lets us determine the 4-th digit after the decimal point of $\lambda^\star$, or the violation of transportation cost constraint is less than 10^{-4}. Note that a violation of 10^{-4} is extremely small, compared with $\delta = \epsilon \sum_{i=1}^n \mathbf{x}_i$, which is much (usually at least 10^5 times) larger than the tolerance since the pixel sum $\sum_{i=1}^n \mathbf{x}_i$ is usually a large number.

In practice, the upper bound (9) and (19) are often between 2 and 3 thanks to gradient normalization. Thus, the bisection method satisfies the stopping criterion in at most 15 iterations ($2 \times 2^{-15} \approx 10^{-4}$).

F.3. Step Sizes of PGD

PGD with projected Sinkhorn On MNIST, CIFAR-10 and ImageNet, the step sizes are set to 0.1. Notice that 0.1 is also the step size used by Wong et al. (2019) on MNIST and CIFAR-10. The gradient is normalized using ℓ_∞ norm:

$$\underset{\|v\|_\infty \leq 1}{\operatorname{argmax}}\, v^\top \nabla_{\mathbf{x}} \ell(\mathbf{x}, y) = \operatorname{sign}\left(\nabla_{\mathbf{x}} \ell(\mathbf{x}, y)\right).$$

Again, this is the same setting used by Wong et al. (2019). While $\eta = 1.0$ achieves lower adversarial accuracy on the first batch of samples in Appendix F.5, we find this large step size causes numerical overflow easily on the remaining batches. Thus we choose $\eta = 0.1$ to present the experimental results.

PGD with dual projection On MNIST, the step size is set to 0.1. On CIFAR-10 and ImageNet, the step size is set to 0.01. The gradient is normalized in the following way:

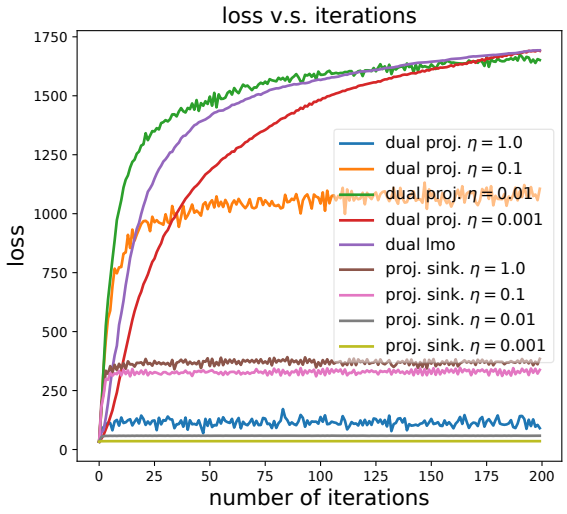
$$\frac{\nabla_\Pi \ell(\Pi^\top \mathbf{1}, y)}{\max_{i,j} |\nabla_\Pi \ell(\Pi^\top \mathbf{1}, y)|}.$$

F.4. Implementation of Local Transportation and Sparse Matrices Computation

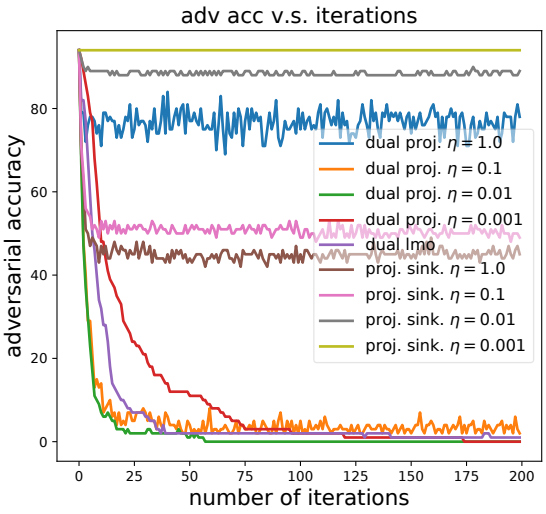
The local transportation technique in §5.1 requires computation on a sparse matrix Π. However, a big challenge is that sparse matrices computation is not easily parallelizable on GPUs. As such, current deep learning packages (PyTorch, TensorFlow) do not support general sparse matrices well.

To fully utilize GPU acceleration, we explore the sparsity pattern in Π. Notice that each row of Π has at most k^2 nonzero entries. We store Π as a $n \times k^2$ dense matrix, with some possible dummy entries. The advantage is that now Π is a dense matrix. Any row operations on Π (*e.g.* softmin along each row, sorting along each row) can be parallelized easily. The downside, however, is that column operations (*e.g.* summation over each column) might take extra efforts. Nevertheless, this is not a bottleneck of the speed of dual projection and dual LMO, since they only require efficient row operations.

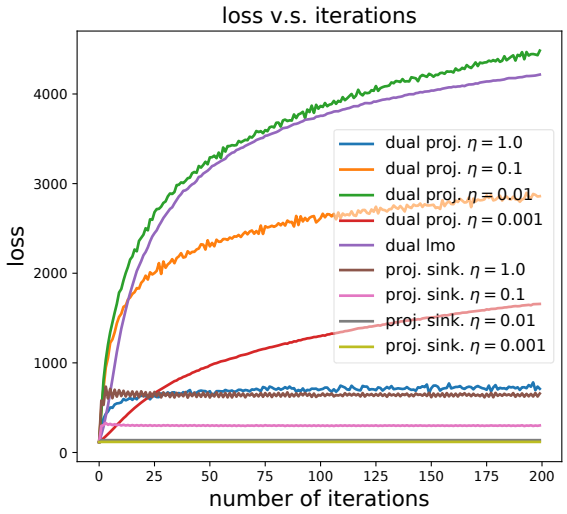
F.5. Further Results on Convergence of Outer Maximization



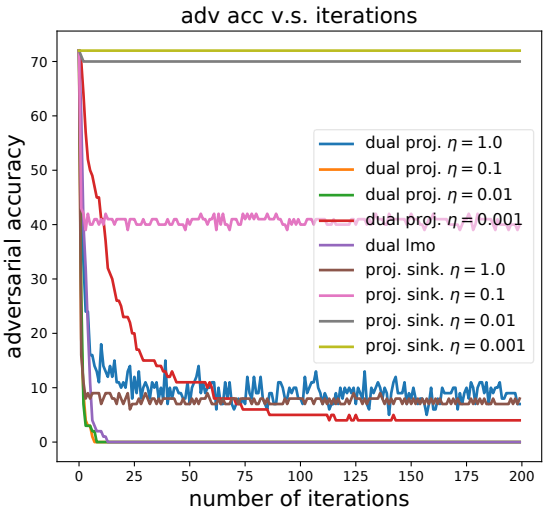
(a) loss w.r.t. iterations on CIFAR-10



(b) adversarial accuracy w.r.t. iterations on CIFAR-10



(c) loss w.r.t. iterations on ImageNet



(d) adversarial accuracy w.r.t. iterations on ImageNet

Figure 8: Convergence of outer maximization of different attacks.

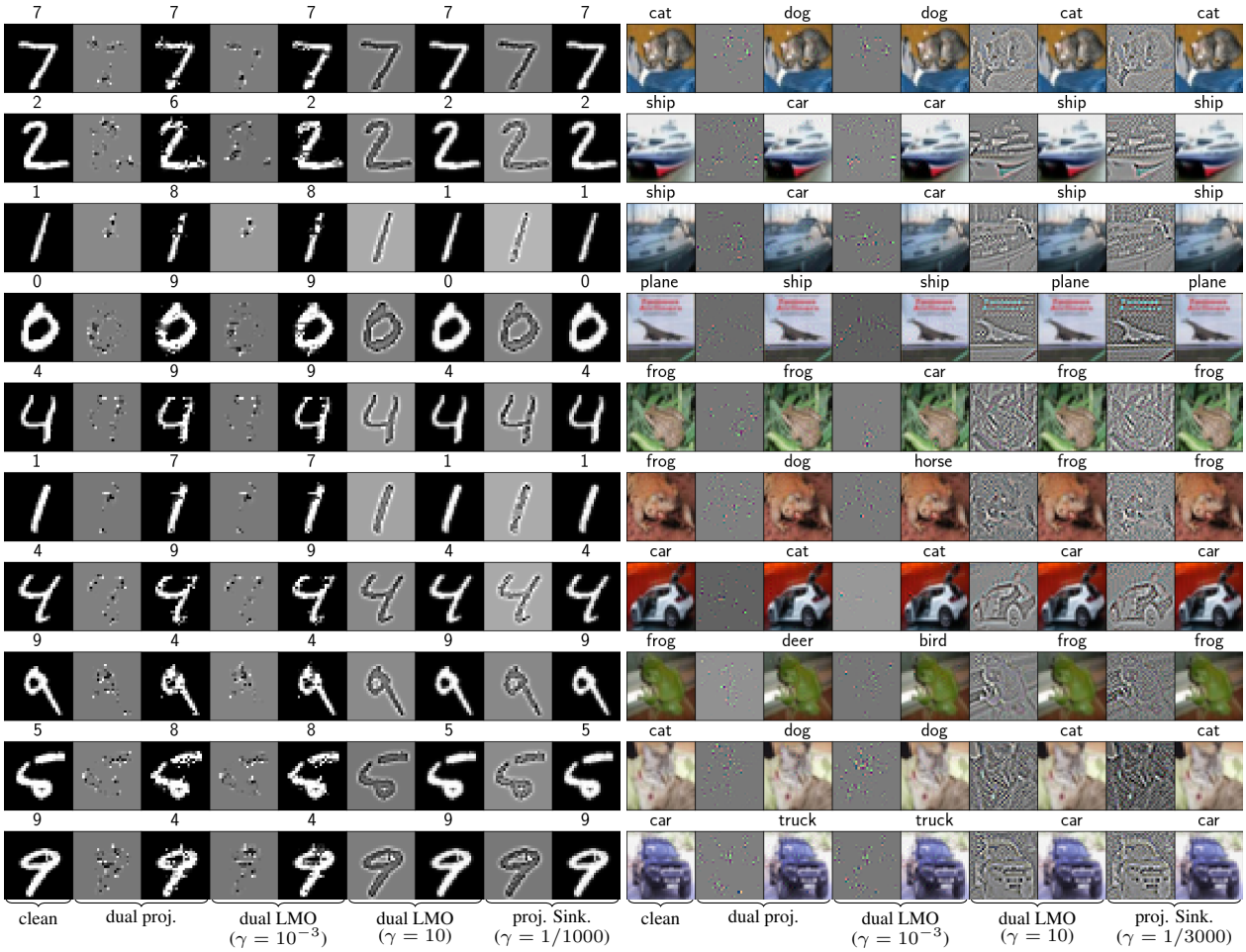
We plot the loss and adversarial accuracy w.r.t. the number of iterations in Figure 8 ($\epsilon = 0.005$ on both CIFAR-10 and ImageNet). Dual LMO uses $\gamma = 10^{-3}$. Projected Sinkhorn uses $\gamma = 5 \cdot 10^{-5}$ on CIFAR-10 and $\gamma = 5 \cdot 10^{-6}$ on ImageNet.

Frank-Wolfe with dual LMO FW uses the default decay schedule $\frac{2}{t+1}$. We observe that FW with dual LMO converges very fast especially at the initial stage, even when using the simple default decay schedule.

PGD with Projected Sinkhorn We observe that when η is small (*e.g.* $\eta = 0.01, 0.001$), PGD with projected Sinkhorn barely makes progress in the optimization. While aggressively large step sizes (*e.g.* 1.0 and 0.1) can make progress, the curves are very noisy indicating the steps sizes are too large, and the loss is still much lower than the other two attacks.

PGD with Dual Projection In contrast, PGD with dual projection has more meaningful curves: small η (*e.g.* 0.001) converges very slowly; large η (*e.g.* 1.0 and 0.1) makes the curves noisy, while appropriate choice of η (*e.g.* 0.01) always achieves the highest loss and also the lowest adversarial accuracy. In these cases, PGD with dual projection converges comparably and sometimes slightly faster than Frank-Wolfe.

F.6. MNIST and CIFAR-10 Adversarial Examples



(a) MNIST (b) CIFAR-10

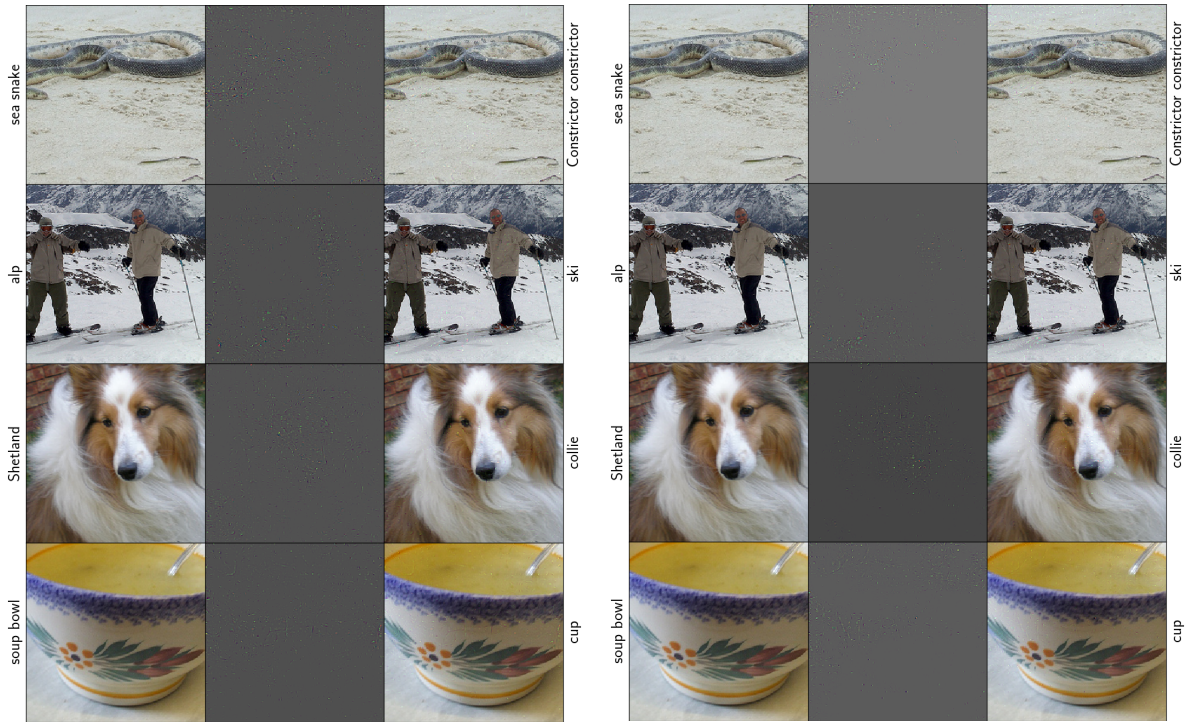
Figure 9: Comparison of Wasserstein adversarial examples and Wasserstein perturbations generated by different attacks on MNIST ($\epsilon = 0.2$) and CIFAR-10 ($\epsilon = 0.002$). Predicted labels are shown on the top of images. Perturbations are scaled linearly to $[0, 1]$ for visualization.

Wasserstein adversarial perturbations generated by PGD with dual projection and FW with dual LMO ($\gamma = 10^{-3}$) are very sparse. Hence, they do not reflect the shapes in the clean images. Perturbations reflect the shapes only when the entropic regularization introduces large approximation error (*e.g.* FW with dual LMO ($\gamma = 10$) and PGD with projected Sinkhorn).

For the sake of visualization of the approximation error, we let PGD use smaller step sizes ($\eta = 0.01$) and more iterations (1000) when combined with projected Sinkhorn.[2] We observe that using large step size, *e.g.*, $\eta = 0.1$, often generates blurry perturbations (not necessarily reflecting shapes clearly). But they are still much more dense compared with adversarial perturbations generated dual projection and dual LMO ($\gamma = 10^{-3}$).

[2]This is the same case for Figure 6 in the main paper. Notice that our normalization method is slightly different from that of Wong et al. (2019), which might account for the slight difference of the visualization results.

F.7. ImageNet Adversarial Examples



(a) Dual projection

(b) Dual LMO ($\gamma = 10^{-3}$)

(c) Dual LMO ($\gamma = 10$)

(d) Projected Sinkhorn ($\gamma = 10^{-4}$)

Figure 10: Comparison of Wasserstein adversarial examples generated by different attacks ($\epsilon = 0.005$). Notice that adversarial pertubations reflect shapes in the images only when the entropic regularization is large (bottom two subfigures).

F.8. Why Does Entropic Regularization Reflect Shapes in Images?

A large entropic regularization term in the optimization objective (of projected Sinkhorn and dual LMO) encourages the transportation to be uniform, thus each pixel tends to spread its mass evenly to its nearby pixels. In the region where pixel intensities do not change much, the transportations cancel out; while in the region where pixel intensities change drastically (*e.g.*, edges in an image), pixel mass flows from the high pixel region to low pixel region. As a result, it reflects the edges in the original image.

F.9. Analysis of Running Time of Projected Sinkhorn

One possible explanation for slow per iteration running speed of projected Sinkhorn is that, each iteration of projected Sinkhorn requires solving n nonlinear univariate equations. In implementation, this step is done by calling Lambert function, which eventually calls Halley's method, a root-finding algorithm. While solving each nonlinear equation is counted as $O(1)$ in analysis, this operation is potentially much more expensive than other basic arithmetic operations (*e.g.* addition and multiplication). In contrast, both dual projection and dual linear minimization only rely on more standard arithmetic operations (*e.g.* multiplication, division and comparison, logarithm and exponential functions).

We test the running time of the nonlinear equation solvers in experiments. On a single RTX 2080 Ti GPU, for a batch of 100 samples on MNIST, we observe that Lambert function evaluation takes about 26% of the running time during each iteration of projected Sinkhorn. For a batch of 100 samples on CIFAR-10, Lambert function evaluation takes about 28% of the running time during each iteration of projected Sinkhorn. For a batch of 50 samples on ImageNet, Lambert function evaluation takes about 32% of the running time of one iteration of projected Sinkhorn.

We note that GPU time recording is somewhat inconsistent on different machines. In our case, all experiments in the main paper are run on a single P100 GPU on a cluster. However, on a 2080 Ti local GPU, we observe that projected Sinkhorn is slightly faster than dual projection on ImageNet in terms of per iteration running time (but still at least twice slower than dual LMO). While on MNIST and CIFAR-10, dual projection and dual LMO are consistently faster than projected Sinkhorn, both on P100 cluster and 2080 Ti local machine.

F.10. Case Study: Feasibility of Generated Wasserstein Adversarial Examples

In this section, we present a sanity check of feasibility of adversarial examples generated by different algorithms on MNIST.

Table 4: A sanity check of feasibility of Wasserstein adversarial examples generated by different algorithms on MNIST. **2nd column:** The average Wasserstein distance between adversarial examples and clean images (the higher the better). **3rd column:** The maximum pixel value in the generated adversarial examples (the lower the better). **4th column:** The percentage of pixel mass that exceeds 1 (the lower the better). The third and the forth columns are largest values over a mini-batch, while the second column is the average over a mini-batch.

method	$\mathcal{W}(\mathbf{x}, \mathbf{x}_{adv})$	$\max_i \{\mathbf{x}_i\}$	$\frac{\sum_{i=1}^n \max\{\mathbf{x}_i - 1, 0\}}{\sum_{i=1}^n \mathbf{x}_i}$
PGD + Proj. Sink. ($\gamma = 1/1000$)	0.109965	1.474048	3.646899%
PGD + Dual Proj. (w/o post-processing)	0.493146	4.118621	15.812986%
PGD + Dual Proj.	0.444885	1.000030	0.000034%
FW + Dual LMO (w/o post-processing)	0.428238	3.955514	10.406417%
FW + Dual LMO	0.399014	1.000015	0.000025%

Setup We test all attacks on the first 100 samples on the test set of MNIST with $\epsilon = 0.5$. All parameter settings are the same as the table in the main paper. Dual LMO uses $\gamma = 10^{-3}$ and projected Sinkhorn uses $\gamma = 1/1000$.

Wasserstein Constraint The (exact) Wasserstein distances presented in Table 4 are calculated by a linear programming solver. We observe that all adversarial examples strictly satisfy the the Wasserstein constraint $\mathcal{W}(\mathbf{x}, \mathbf{x}_{adv}) \leq 0.5$. We also observe that the Wasserstein adversarial examples generated by PGD with dual projection and FW with dual LMO have much larger Wasserstein distance than those of projected Sinkhorn. This again hightlights that projected Sinkhorn is only an approximate projection operator, thus PGD cannot fully explore the Wasserstein ball, resulting in a weak attack.

Hypercube Constraint Without post-processing, all attacks generate Wasserstein adversarial examples that violate the hypercube constraint a lot. However, after applying our post-processing algorithm, the generated Wasserstein adversarial examples roughly satisfy the hypercube constraint $[0, 1]^n$ up to a reasonable precision.

F.11. Adversarially Trained Models

In this section, we present additional experiments on adversarial training and attacking adversarially trained models.

F.11.1. MNIST

We adversarially train a robust model using Frank-Wolfe with dual LMO and a fixed perturbation budget $\epsilon = 0.3$. The inner maximization is approximated with 40 iterations of Frank-Wolfe.[3] This model achieves 95.83% clean accuracy. The adversarial accuracy of this model is shown in Table 5.

For comparison, we also present results on attacking an adversarially trained model by PGD with projected Sinkhorn. We use a pretrained model released by Wong et al. (2019), which is adversarially trained using an adaptive perturbation budget $\epsilon \in [0.1, 2.1]$. This model achieves 97.28% clean accuracy.

We notice that the model adversarially trained by PGD with projected Sinkhorn seems to overfit to the same attack. Compared with the standard trained model in Table 3, the adversarially trained model has higher adversarial accuracy under projected Sinkhorn, but has lower adversarial accuracy under our stronger attacks.

In Table 6, the post-processing algorithm does not work quite well with Frank-Wolfe. Normally, increasing the perturbation budget ϵ should decrease the adversarial accuracy. Indeed, if we do not post-process the output, the adversarial accuracy decreases monotonically for Frank-Wolfe. However, after post-processing, the adversarial accuracy increases a lot, and even increases as the the perturbation budget increases. For this model, our post-processing algorithm does not seems to be "compatible" with Frank-Wolfe. Doing a better job on optimizing the Wasserstein constrained problem ignoring the hypercube constraint does not necessarily give a good solution to the problem with hypercube constraint.

Table 5: MNIST model adversarially trained by Frank-Wolfe with dual LMO ($\epsilon = 0.3$).

method	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$	$\epsilon = 0.4$	$\epsilon = 0.5$
PGD + Proj. Sink. ($\gamma = 1/1000$)	94.9	94.0	93.0	91.9	90.5
PGD + Proj. Sink. ($\gamma = 1/1500$)	94.5	93.2	91.5	89.3	86.8
PGD + Proj. Sink. ($\gamma = 1/2000$)	–	–	–	–	–
PGD + Dual Proj.	92.6	87.8	80.2	70.7	59.7
FW + Dual LMO	92.5	88.1	82.1	75.3	66.8
PGD + Dual Proj. (w/o post-processing)	91.1	82.9	71.3	58.4	44.6
FW + Dual LMO (w/o post-processing)	91.1	83.9	73.9	63.0	50.9

Table 6: MNIST model adversarially trained by PGD with projected Sinkhorn.

method	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$	$\epsilon = 0.4$	$\epsilon = 0.5$
PGD + Proj. Sink. ($\gamma = 1/1000$)	95.0	92.4	90.5	88.5	86.5
PGD + Proj. Sink. ($\gamma = 1/1500$)	93.8	90.0	82.7	85.2	90.5
PGD + Proj. Sink. ($\gamma = 1/2000$)	–	–	–	–	–
PGD + Dual Proj.	1.1	0.8	0.6	0.6	0.6
FW + Dual LMO	4.8	21.6	34.5	39.2	39.6
PGD + Dual Proj. (w/o post-processing)	1.0	0.4	0.3	0.3	0.3
FW + Dual LMO (w/o post-processing)	0.8	0.3	0.2	0.1	0.0

[3] We also have tried larger perturbation budgets $\epsilon = 0.4$ and $\epsilon = 0.5$, but the training collapses.

F.11.2. CIFAR-10

We adversarially train a robust model using Frank-Wolfe with dual LMO and a fixed perturbation budget $\epsilon = 0.005$. The inner maximization is approximated with at most 30 iterations of Frank-Wolfe. This model achieves 82.57% clean accuracy. The adversarial accuracy of this model is shown in Table 7.

For comparison, we also present results on attacking an adversarially trained model by PGD with projected Sinkhorn. We use a pretrained model released by Wong et al. (2019), which is adversarially trained using an adaptive perturbation budget $\epsilon \in [0.01, 0.38]$. This model achieves 81.68% clean accuracy.

Table 7: CIFAR-10 model adversarially trained by Frank-Wolfe with dual LMO ($\epsilon = 0.005$)

method	$\epsilon = 0.001$	$\epsilon = 0.002$	$\epsilon = 0.003$	$\epsilon = 0.004$	$\epsilon = 0.005$
PGD + Proj. Sink. ($\gamma = 1/3000$)	82.4	82.3	82.1	81.9	81.8
PGD + Proj. Sink. ($\gamma = 1/10000$)	82.0	81.5	81.0	80.6	80.1
PGD + Proj. Sink. ($\gamma = 1/20000$)	–	–	–	–	–
PGD + Dual Proj.	76.8	71.8	67.0	62.0	56.8
FW + Dual LMO	77.4	73.7	70.2	66.8	62.6
PGD + Dual Proj. (w/o post-processing)	76.5	71.3	66.4	61.4	56.2
FW + Dual LMO (w/o post-processing)	77.2	73.7	70.4	67.3	63.9

Table 8: CIFAR-10 model adversarially trained by PGD with projected Sinkhorn.

method	$\epsilon = 0.001$	$\epsilon = 0.002$	$\epsilon = 0.003$	$\epsilon = 0.004$	$\epsilon = 0.005$
PGD + Proj. Sink. ($\gamma = 1/3000$)	81.7	81.6	81.6	81.6	81.5
PGD + Proj. Sink. ($\gamma = 1/10000$)	81.6	81.4	81.3	81.2	81.0
PGD + Proj. Sink. ($\gamma = 1/20000$)	–	–	–	–	–
PGD + Dual Proj.	72.5	64.4	56.3	48.8	42.2
FW + Dual LMO	72.4	64.0	55.5	47.8	41.2
PGD + Dual Proj. (w/o post-processing)	72.0	63.3	54.9	47.2	40.6
FW + Dual LMO (w/o post-processing)	71.7	62.2	52.9	44.7	37.5

G. Post-processing: Capacity Constrained Projection for Hypercube Constraint

In this section, we present the post-processing algorithm mentioned in §5.3.

G.1. Algorithm

Suppose that pixel values of input images are represented by real numbers in $[0,1]^n$. We need to add one additional constraint $\Pi^\top \mathbf{1} \le \mathbf{1}$, which results in the following Euclidean projection problem:

$$\begin{aligned} &\underset{\Pi \ge 0}{\text{minimize}} \quad \frac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 \\ &\text{subject to } \Pi\mathbf{1} = \mathbf{x},\ \Pi^\top \mathbf{1} \le \mathbf{1}, \langle \Pi, C\rangle \le \delta. \end{aligned} \tag{28}$$

We call this problem a *capacity constrained projection*, since the additional constraint essentially specifies the maximum mass that a pixel location can receive. We introduce the partial Lagrangian to derive the following dual problem:

$$\underset{\lambda \ge 0, \boldsymbol{\mu} \ge 0}{\text{maximize}}\ g(\lambda, \boldsymbol{\mu}),$$

where

$$\begin{aligned} g(\lambda, \boldsymbol{\mu}) &= \min_{\Pi \ge 0, \Pi\mathbf{1} = \mathbf{x}} \frac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 + \lambda\left(\langle \Pi, C\rangle - \delta\right) + \boldsymbol{\mu}^\top\left(\Pi^\top\mathbf{1} - \mathbf{1}\right) \\ &= \min_{\Pi \ge 0, \Pi\mathbf{1} = \mathbf{x}} \frac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 + \lambda\left(\langle \Pi, C\rangle - \delta\right) + \langle \Pi, \mathbf{1}\boldsymbol{\mu}^\top\rangle - \boldsymbol{\mu}^\top\mathbf{1} \\ &= \min_{\Pi \ge 0, \Pi\mathbf{1} = \mathbf{x}} \frac{1}{2}\|\Pi - G + \lambda C + \mathbf{1}\boldsymbol{\mu}^\top\|_{\mathrm{F}}^2 - \langle G, \lambda C + \mathbf{1}\boldsymbol{\mu}^\top\rangle + \frac{1}{2}\|\lambda C + \mathbf{1}\boldsymbol{\mu}^\top\|_{\mathrm{F}}^2 - \lambda\delta - \boldsymbol{\mu}^\top\mathbf{1} \end{aligned}$$

We optimize $g(\lambda, \boldsymbol{\mu})$ by alternating maximization on λ and $\boldsymbol{\mu}$ respectively: fixing $\boldsymbol{\mu}$, we maximize λ via bisection method; fixing λ, we maximize $\boldsymbol{\mu}$ via k steps gradient ascent with nesterov acceleration, where k is a hyperparameter. Evaluating the gradient of $\boldsymbol{\mu}$ in bisection method, as well as evaluating the gradient of $\boldsymbol{\mu}$, can be reduced to simplex projections following similar derivations in dual projection (§3).

Due to sublinear convergence rate of gradient ascent, it may be slow to obtain a very high precision solution. However, empirical evidences show that a few hundred alternating maximization (with k around 10 or 20) converges to a solution with reasonable precision (*e.g.* satisfying hypercube constraint up to the third digit after the decimal point), and convergence to these modest precision solutions is already sufficient for generating valid Wasserstein adversarial examples.

H. Additional Related Work

In this section, we comment on a few more works related to threat models beyond the standard ℓ_p metric. A number of authors have recently explored geometric transformations as an adversarial attack against deep models. As already mentioned in the main paper, (Engstrom et al., 2019) studied adversarial rotations and translations where perturbation is measured by the degree of rotation and translation. Similarly, Alaifari et al. (2019) Alaifari et al. (2019) considered adversarial deformations and used the maximum Euclidean size of the deformation as the perturbation budget. Kanbak et al. (2018) studied the more general case where spatial transformation is parameterized as a Lie group, and employed the geodesic distance in the image appearance manifold to measure perturbation size. Xiao et al. (2018), on the other hand, modeled spatial transformations as a vector flow and used the total variation of the flow to measure perturbation size. On a high level, spatial transformation shares some similarity with the Wasserstein threat model, as they both involve pixel mass movement. In fact, the Wasserstein threat model can be treated as a further relaxation of spatial transformations, where we are not only allowed to move pixels but also to change pixel values (*e.g.* pixel mass splitting). In this sense, the Wasserstein threat model is a combination of spatial transformation and the standard ℓ_p additive perturbation, although the way it measures perturbation is entirely different from either. All of the above methods employ first order gradient algorithms to solve their respective optimization problems. We note that some authors, *e.g.* Athalye et al. (2018); Li et al. (2019), have already considered physical attack for real world objects.

Lastly, we mention that certification algorithms for spatial transformations and the Wasserstein threat model have recently been developed by Balunovic et al. (2019) and Levine & Feizi (2019), respectively.

Additional References

Alaifari, R., Alberti, G. S., and Gauksson, T. ADef: An Iterative Algorithm to Construct Adversarial Deformations. In *International Conference on Learning Representations*, 2019.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing Robust Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 284–293, 2018.

Balunovic, M., Baader, M., Singh, G., Gehr, T., and Vechev, M. Certifying Geometric Robustness of Neural Networks. In *Advances in Neural Information Processing Systems 32*, pp. 15313–15323, 2019.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. Exploring the Landscape of Spatial Robustness. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1802–1811, 2019.

Kanbak, C., Moosavi-Dezfooli, S.-M., and Frossard, P. Geometric Robustness of Deep Networks: Analysis and Improvement. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Levine, A. and Feizi, S. Wasserstein Smoothing: Certified Robustness against Wasserstein Adversarial Attacks. In *The 23rd International Conference on Artificial Intelligence and Statistics*, 2019.

Li, J., Schmidt, F., and Kolter, Z. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3896–3904, 2019.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. Spatially Transformed Adversarial Examples. In *International Conference on Learning Representations*, 2018.